User guide for QSDP-0 – a MATLAB software package for convex quadratic semidefinite programming

Kim-Chuan Toh *

February 25, 2010

Working paper

Abstract

This software is designed to solve a convex quadratic semidefinite programming (QSDP) problem, possibly with a log-determinant term. It employs an infeasible primal-dual predictor-corrector path-following method using the Nesterov-Todd search direction. The basic code is written in MATLAB, but key subroutines in C are incorporated via Mex interface. It also uses functions in the software for linear semidefinite programming, SDPT3-3.1. Here we briefly describe how to install and run QSDP-0. We should emphasize that the current version is an experimental software and it is not intended to be a general purpose solver. Some numerical results are presented to illustrate the performance of the software on QSDPs arising from the nearest correlation matrix and the Euclidean distance matrix completion problems.

1 Introduction

Let $\mathcal{S}^n$ be the space of real $n \times n$ symmetric matrices, and S^n_+ the cone of $n \times n$ symmetric positive semidefinite matrices. The software package, QSDP-0, is designed to solve a convex quadratic semidefinite programming (QSDP) problem of the following form:

$$\min_X \Big\{ \frac{1}{2} X \bullet \mathcal{Q}(X) + C \bullet X - \beta \log\det(X) \,:\, \mathcal{A}(X) = b, \quad X \succeq 0 \Big\}, \tag{1}$$

where $\beta \geq 0$ is a given parameter, $\mathcal{Q} : \mathcal{S}^n \to \mathcal{S}^n$ is a given self-adjoint positive semidefinite linear operator, $\mathcal{A} : \mathcal{S}^n \to I\!R^m$ is a linear map, and $b \in I\!R^m$. The

*Department of Mathematics, National University of Singapore, Blk S17 (SOC1), 10 Lower Kent Ridge Road, Singapore 119076 (mattohkc@nus.edu.sg); and Singapore-MIT Alliance, 4 Engineering Drive 3, Singapore 117576.

notation $X \succeq 0$ means that $X \in \mathcal{S}^n_+$. The linear map $\mathcal{A}$ can explicitly be written as:

$$\mathcal{A}(X) = \begin{bmatrix} A_1 \bullet X \\ \vdots \\ A_m \bullet X \end{bmatrix},$$

where $A_1, \ldots, A_m \in \mathcal{S}^n$ are given constraint matrices. Let $\mathcal{A}^T : I\!R^m \to \mathcal{S}^n$ be the adjoint of $\mathcal{A}$, which is defined by $\mathcal{A}^T y = \sum_{k=1}^m y_k A_k$. The Lagrangian dual problem of (1) is given by:

$$\begin{array}{ll} \max\limits_{X,y,Z} & -\frac{1}{2} X \bullet \mathcal{Q}(X) + b^T y + \beta \log\det(Z) + \beta n(1 - \log\beta) \\ \text{s.t.} & \mathcal{A}^T(y) - \mathcal{Q}(X) + Z = C,\ Z \succeq 0. \end{array} \tag{2}$$

Caveats. Our algorithm needs to store the primal iterate X, which is typically dense. Thus our software may not be able to handle a QSDP with matrix dimension $n \geq 3000$ on an average PC available today. We also assume that the number of linear constraints m in (1) is less than a few thousands, say $m \leq 5000$, so that a matrix of form $\mathcal{A}(U \circledast V)\mathcal{A}^T$ can be computed and stored explicitly. Here $U \circledast V$ is the symmetrized Kronecker product of the symmetric matrices U, V.

It is well-known that (1) can be reformulated into a standard semidefinite-quadratic-linear program (SQLP) by modeling the quadratic term in the objective function by a second order cone constraint together with q extra linear equality constraints, where q is the rank of $\mathcal{Q}$. As the resulting SQLP problem has at least $m + q$ linear equality constraints, and the computational cost of a standard interior-point method for solving such a problem is at least $\Theta(m+q)^3$, it is not difficult to see that reformulating (1) as an SQLP would be computational undesirable when $q = O(n^2)$. However, for $n \leq 100$, the SQLP reformulation of (1) can be quite attractive since there are efficiently and robust interior-point methods for solving a standard SQLP.

There is another form of quadratic SDP given by $\min_y\{y^T H y/2 + b^T y : \mathcal{A}^T y \preceq C,\ y \in I\!R^m\}$, where H is a given positive semidefinite matrix. For this problem, the Schur complement matrix arising at each interior-point iteration has the form $H + M$ (with M being the Schur complement matrix associated with the linear SDP without the quadratic term), and the computation involved at each iteration is very similar to that for a standard linear SDP. As our interest is in problems with quadratic terms involving the matrix variables, we shall not consider this form of quadratic SDP in this software.

For later discussion, we state an example of QSDP arising from estimating the nearest correlation matrix (NCM) of a given data matrix $K \in S^n$:

$$\min_X \Big\{ \frac{1}{2} \|\mathcal{L}(X - K)\|_F^2 \ :\ \mathrm{diag}(X) = e, X \succeq 0 \Big\}, \tag{3}$$

where e is the vector of ones, and $\mathcal{L} : \mathcal{S}^n \to \mathcal{S}^n$ is a given linear map. By ignoring the constant term, it is easily seen that (3) can be written in the form (1) with $\mathcal{Q} = \mathcal{L}^T\mathcal{L}$

and $C = -\mathcal{Q}(K)$. There are two popular choices of $\mathcal{Q}$ [7] for the NCM problem: (a) $\mathcal{Q}(X) = (H \circ H) \circ X$ corresponding to $\mathcal{L}(X) = H \circ X$, where $H \in \mathcal{S}^n$ is a given non-negative matrix and "$\circ$" denotes elementwise product; (b) $\mathcal{Q}(X) = UXU$ corresponding to $\mathcal{L}(X) = U^{1/2}XU^{1/2}$, where $U \in S^n$ is a given positive semidefinite matrix.

Another example of QSDP comes from the Euclidean distance matrix (EDM) completion problem.

2 Installation

The current version is written in MATLAB 7.3. It is available from the internet site:

`http://www.math.nus.edu.sg/~mattohkc/qsdp.html`

Our software uses a number of Mex routines generated from C programs written to carry out certain operations that MATLAB is not efficient at. To install QSDP-0 and generate these Mex routines, the user can simply follow the steps below:

(a) unzip QSDP-0.zip;

(b) run MATLAB in the directory QSDP-0;

(c) run the m-file `Installmex.m`.

To check whether QSDP-0 has been installed correctly, the user can run the m-files:

```
>> startup
>> qsdpdemo
```

3 The algorithm

The algorithm implemented in QSDP-0 is an infeasible primal-dual Mehrotra-type predictor-corrector path-following algorithm, described in detail in Appendix A. At each iteration, we first compute a *predictor* search direction aimed at decreasing the duality gap by as much as possible. After that, the algorithm generates a Mehrotra-type corrector step [9] with the intention of keeping the iterates close to the central path. At each iteration with current iterate (X, y, Z), the predictor direction $(\delta X, \delta y, \delta Z)$ is computed from the following linear system:

$$\begin{bmatrix} -W^{-1} \circledast W^{-1} - \mathcal{Q} & \mathcal{A}^T \\ \mathcal{A} & 0 \end{bmatrix} \begin{bmatrix} \delta X \\ \delta y \end{bmatrix} = \begin{bmatrix} R_d - R_c \\ R_p \end{bmatrix}, \tag{4}$$

$$\delta S \;=\; R_d - \mathcal{A}^T \delta y + \mathcal{Q}(\delta X), \tag{5}$$

where $W \succ 0$ is the unique matrix satisfying $WZW = X$ (W is known as the Nesterov-Todd scaling matrix), and

$$R_p \;=\; b - \mathcal{A}(X),\; R_d \;=\; C - Z - \mathcal{A}^T y + \mathcal{Q}(X),\; R_c \;=\; \max\{\sigma\mu(X,Z), \beta\}X^{-1} - Z. \tag{6}$$

Here $\sigma \in (0,1)$ is a centering parameter, and

$$\mu(X,Z) = \frac{1}{n}(X \bullet Z - \beta \log\det(XZ) - \beta n(1 - \log\beta)). \tag{7}$$

The corrector direction $(\Delta X, \Delta y, \Delta Z)$ is also computed from the same linear system but with R_c in (4) appropriately replaced.

The dimension of the linear system (4) is $m + n(n+1)/2$. Typically this system is too large for solution via a direct solver when n is larger than 100. In this software, we use a preconditioned symmetric quasi-minimal residual (PSQMR) iterative method [3] to solve such a system. In [14], three different choices of preconditioners are designed and analyzed, and their relative performance are described in [14].

3.1 Preconditioners used for solving (4)

4 The main function: `qsdp.m`

The main routine that corresponds to Algorithm QSDP-IPC described in Appendix A is `qsdp.m`, whose calling syntax is as follows:

```
[obj,X,y,Z,info,runhist] = qsdp(blk,At,C,b,Q,beta,OPTIONS,X0,y0,Z0).
```

Input arguments.

`blk`: a cell array describing the block structure of the QSDP problem.

`At`, `C`, `b`, `Q`: QSDP data.

`beta`: parameter for the log-determinant term in (1).

If it is not supplied, `beta` is assumed to be 0.

`OPTIONS`: a structure array of parameters (optional).

`X0`, `y0`, `Z0`: an initial iterate (optional).

If the input argument `OPTIONS` is omitted, default values specified in the function `qsdparameters.m` are used. More detail on `OPTIONS` is given in Section 4.1.

Output arguments.

The names chosen for the output arguments explain their contents. Note that `X,Z` are cell arrays such that the contents of `X{1},Z{1}` are approximately optimal primal and dual solutions, respectively, when the corresponding problems are feasible. The reason for using cell arrays `X,Z` to store the matrix variables X, Z is because such a data structure is more flexible for future extension to QSDP problems with multiple positive semidefinite matrix variables.

The argument `info` is a structure array containing performance information such as

```
info.termcode, info.obj,     info.gap,
info.pinfeas,  info.dinfeas, info.cputime
```

whose meanings are explained in `qsdp.m`. The argument `runhist` is a structure array which records the history of various performance measures during the run; for example, `runhist.gap` records the complementarity gap, $n\mu(X, Z)$, at each interior-point iteration.

Note that, while the output (X, y, Z) normally gives approximately optimal solutions, if `info.termcode` is 1 the problem is suspected to be primal infeasible and $(0, y, Z)$ is an approximate certificate of infeasibility, with $b^T y = 1$, $Z \succ 0$, and $\|\mathcal{A}^T y + Z\|_F$ small, while if `info.termcode` is 2 the problem is suspected to be dual infeasible and X is an approximate certificate of infeasibility, with $C \bullet X = -1$, $X \succ 0$, and $\max\{\|\mathcal{A}(X)\|_2, \|\mathcal{Q}(X)\|_F\}$ small.

4.1 The structure array `OPTIONS` for parameters

`qsdp.m` uses a number of parameters which are specified in a MATLAB structure array called `OPTIONS` in the m-file `qsdparameters.m`. If desired, the user can change the values of these parameters. The meaning of a few of the specified fields in `OPTIONS` are given below:

```
OPTIONS.gaptol     = accuracy tolerance
OPTIONS.maxit      = maximum number of interior-point iterations allowed
OPTIONS.psqmrmaxit = maximum number of PSQMR steps allowed per linear system
OPTIONS.precond    = type of preconditioner to use when solving linear system
```

As an example, if the user wants to solve the QSDP problem to an accuracy tolerance of `1e-4` instead of the default value of `1e-6` while using the default values for all other parameters, he only needs to set `OPTIONS.gaptol = 1e-4`.

4.2 Stopping criteria

At a given iterate (X, y, Z), the algorithm QSDP-IPC is stopped when any of the following cases occur.

1. solutions with the desired accuracy have been obtained, i.e.,

$$\phi := \max\{\texttt{relgap}, \texttt{pinfeas}, \texttt{dinfeas}\} \ \leq \ \texttt{OPTIONS.gaptol}, \tag{8}$$

   where

$$\texttt{relgap} = \frac{n\mu(X,Z)}{1+|\texttt{pobj}|+|\texttt{dobj}|}, \tag{9}$$

$$\texttt{pinfeas} = \frac{\|R_p\|_2}{1+\|b\|_2}, \quad \texttt{dinfeas} = \frac{\|R_d\|_F}{1+\|C\|_F}. \tag{10}$$

   Here, "`pobj`" and "`dobj`" denote the primal and dual objective values, and R_p, R_d are defined as in (10).

2. primal infeasibility is suggested because

$$b^T y / \|\mathcal{A}^T y + Z\|_F > 10^8;$$

3. dual infeasibility is suggested because

$$-C \bullet X / \max\{\|\mathcal{A}(X)\|_2, \|\mathcal{Q}(X)\|_F\} > 10^8;$$

4. slow progress is detected, measured by a rather complicated set of tests including

$$\texttt{relgap} < \max\{\texttt{pinfeas}, \texttt{dinfeas}\} ;$$

5. numerical problems are encountered, such as the iterates not being positive definite or the Schur complement matrix not being positive definite; or
6. the step sizes fall below 10^{-6};
7. the number of PSQMR steps required to solve the linear system (4) exceeds the maximum specified in `OPTIONS.psqmrmaxit`.

5 Coding the problem data

The data structure we adopted for QSDP-0 follows that of the linear SDP software, SDPT3-3.1.

To be consistent with SDPT3-3.1, the user needs to specify the block structure of the QSDP problem through the 1×2 cell array `blk`, with

```
blk{1,1} = 's';  blk{1,2} = n;
```

where `'s'` indicates that the constraint cone is $\mathcal{S}^n_+$ and "`n`" is the dimension. The constraint matrices $A_1, \ldots, A_m$ are stored in an $1 \times m$ cell array `At`, where `At{1,k}` = A_k, $k = 1, \ldots, m$. The data $C \in \mathcal{S}^n$ is stored in a cell array as `C{1}` $= C$, and $b \in I\!R^m$ is stored as the usual MATLAB vector. As an example, the data `blk`, `At`, `b` for the QSDP problem in (3) can be coded as follows:

```
blk{1,1} = 's'; blk{1,2} = n;
At = cell(1,n);
for k = 1:n; At{1,k} = spconvert([k,k,1; n,n,0]); end
b = ones(n,1);
```

The user also needs to specify the linear map $\mathcal{Q}$ through a MATLAB structure array `Q` by setting `Q.QXfun` to be the name of the MATLAB function in which $\mathcal{Q}(X)$ is evaluated given X. Any auxiliary variables needed in `Q.QXfun` can be stored in `Q`. As an example, we consider the QSDP in (3) arising from the nearest correlation matrix problem with $\mathcal{Q}(X) = H \circ X$. Suppose the latter is evaluated in the function `QXHadamard.m` (supplied in the software) given by

```
function QX = QX_Hadamard(blk,Q,X)
  QX = Q.mat{1}.*X{1};
```

In this case, we set `Q.QXfun = 'QX_Hadamard'; Q.mat{1} =` H. For the linear map $\mathcal{Q}(X) = UXU$, the user can set `Q.QXfun = 'QX_SKP'; Q.mat{1} =` U.

Important note: the calling syntax for `Q.QXfun` must have the form as shown in `QX_Hadamard.m`.

6 Examples

We will now generate some sample runs to illustrate how our package might be used to solve some test problems. We assume that the current directory is QSDP-0.

```
>> startup     %% set up Matlab paths
>> n=50; m=100;
>> [blk,At,C,b,Q] = randQSDP(n,m); %% generate a random QSDP
>> [obj,X,y,Z,info,runhist] = qsdp(blk,At,C,b,Q);

 qsdp: converting At into required format
 Qnorm = 1.00e+00
 num. of constraints = 100
 dim. of sdp    var  = 50,   num. of sdp  blk  =  1
*****************************************************************************
   qsdp: Inexact primal-dual path-following algorithms
*****************************************************************************
 version  predcorr  gam   precond         QXfun    kappa
   NT       1      0.000  constrained  QXalphaI  1.00e-02

it  pstep dstep p_infeas d_infeas  gap     mean(obj)   cputime  r  psqmr
---------------------------------------------------------------------------
 0  0.000 0.000 1.9e+01 1.5e+01  5.8e+06  1.686322e+06  0:0:00 |%|  5  5
 1  0.913 0.913 1.6e+00 1.3e+00  6.1e+05  1.455295e+05  0:0:01 |%|  7  9
 2  1.000 1.000 8.2e-12 3.7e-16  1.7e+05  5.217435e+04  0:0:01 |%| 17 16
 3  0.917 0.917 1.3e-11 4.1e-16  2.3e+04  3.948290e+04  0:0:02 |%| 33 23
 4  1.000 1.000 5.0e-12 4.6e-16  2.9e+03  3.923376e+04  0:0:03 |%| 55 32
 5  1.000 1.000 6.4e-13 4.8e-16  1.4e+03  3.905145e+04  0:0:05 |$|  5  5
 6  0.972 0.972 1.0e-07 4.6e-16  4.6e+01  3.892347e+04  0:0:05 |$|  5  5
 7  0.973 0.973 5.9e-08 4.6e-16  1.7e+00  3.891905e+04  0:0:06 |$|  5  5
 8  0.972 0.972 2.4e-08 4.4e-16  5.6e-02  3.891882e+04  0:0:07 |$|  5  5
 9  0.967 0.967 9.4e-10 4.7e-16  2.0e-03  3.891881e+04  0:0:07
 Stop: max(relative gap, infeasibilities) < 1.00e-06
-----------------------------------------------------
 number of iterations   =  9
 primal objective value =  3.89188089e+04
 dual   objective value =  3.89188069e+04
 gap := trace(XZ)       = 1.96e-03
 relative gap           = 5.03e-08
```

```
 actual relative gap    = 5.06e-08
 rel. primal infeas     = 9.36e-10
 rel. dual   infeas     = 4.65e-16
 Total CPU time (secs)  = 7.3
 CPU time per iteration = 0.8
 norm(X),norm(u),norm(S) = 2.06e+02, 1.71e+01, 5.92e+02
-------------------------------------------------
```

The meaning of the columns in the displayed output are as follows:

```
it      = interior-point iteration count
pstep   = step-length taken for primal variable
dsetp   = step-length taken for dual variable
pinfeas = relative primal infeasibility, see (10)
dinfeas = relative dual infeasibility, see (10)
gap     = complementarity gap, as defined in the numerator of (9)
mean(obj) = mean of the primal and dual objective values
cputime   = cumulative CPU time taken
r         = type of preconditioner used, or the number of small eigenvalues of Z
psqmr     = number of PSQMR steps taken to solve the linear systems
            at each iteration
```

In the next example, we generate a sample run for a QSDP arising from the nearest correlation matrix problem (3). The reader is referred to the m-file `NCMexample.m` for the details.

```
>> NCMexample('NCM100-Randcorr-1','Hadamard');

 0.5*norm(Q(X-B),'fro')^2 based on simple projection = 2.00e+01

 qsdp: converting At into required format
 Qnorm = 5.52e+00
 num. of constraints = 100
 dim. of sdp    var  = 100,   num. of sdp  blk  =  1
*******************************************************************
   qsdp: Inexact primal-dual path-following algorithms
*******************************************************************
 version  predcorr  gam   precond         QXfun    kappa
   NT       1      0.000  constrained  QXHadamard  1.00e-02

it  pstep dstep p_infeas d_infeas  gap     mean(obj)   cputime  r  psqmr
-------------------------------------------------------------------------
 0  0.000 0.000 6.3e+01 3.1e+01  8.6e+04 -1.594707e+04  0:0:00 |%| 10  7
 1  0.964 0.964 2.3e+00 1.1e+00  4.1e+03 -1.093712e+03  0:0:01 |%|  5  6
 2  1.000 1.000 9.5e-08 1.3e-16  4.5e+02 -1.509343e+02  0:0:01 |%| 10  9
 3  0.971 0.971 5.6e-08 8.0e-17  7.0e+01 -4.971453e+00  0:0:01 |%| 14 13
```

```
 4  0.990 0.990 1.1e-08 7.4e-17  9.7e+00  1.082545e+01  0:0:02 |%| 29 26
 5  0.972 0.972 3.5e-09 7.4e-17  1.1e+00  1.202468e+01  0:0:03 |%|. 107 86
 6  0.949 0.949 7.7e-10 7.4e-17  9.0e-02  1.208213e+01  0:0:05 |$| 10  8
 7  0.919 0.919 6.0e-07 8.0e-17  9.7e-03  1.208395e+01  0:0:05 |$| 11  9
 8  0.774 0.774 1.1e-07 7.7e-17  2.2e-03  1.208418e+01  0:0:06 |$| 13 13
 9  0.992 0.992 2.9e-10 8.2e-17  5.2e-04  1.208416e+01  0:0:07 |$| 13 11
10  0.966 0.966 8.5e-10 8.9e-17  2.0e-05  1.208421e+01  0:0:08 |$| 17 14
11  0.977 0.977 2.8e-11 8.0e-17  2.1e-06  1.208421e+01  0:0:09
 Stop: max(relative gap, infeasibilities) < 1.00e-06
-----------------------------------------------------
 number of iterations   = 11
 primal objective value =  1.20842082e+01
 dual   objective value =  1.20842061e+01
 gap := trace(XZ)       = 2.08e-06
 relative gap           = 1.59e-07
 actual relative gap    = 1.59e-07
 rel. primal infeas     = 2.83e-11
 rel. dual   infeas     = 8.02e-17
 Total CPU time (secs)  = 8.6
 CPU time per iteration = 0.8
 norm(X),norm(u),norm(S) = 2.11e+01, 7.69e+00, 1.27e+01
-----------------------------------------------------
 0.5*norm(Q(X-B),'fro')^2 based on QSDP = 1.208e+01
-----------------------------------------------------
```

Appendix: reformulating QSDP as an SQLP

As mentioned in the Introduction, one can reformulate (1) as a standard SQLP, we show how this is done here. First, factorize $\mathcal{Q}$ as $\mathcal{Q} = \mathcal{R}^T\mathcal{R}$. Then $\langle X, \mathcal{Q}(X)\rangle \le t$ is equivalent to the constraints: $\|\mathcal{R}(X)\| \le s$ and $s^2 \le t$. Hence (1) can equivalently be written as follows:

$$\begin{array}{ll} \min_X & \frac{1}{2}t + C \bullet X \\ \text{s.t.} & \mathcal{A}(X) = b, \quad X \succeq 0 \\ & \mathcal{R}(X) - z = 0, \quad \|z\| \le s \\ & s^2 \le t. \end{array}$$

The standard SQLP reformulation of (1) then becomes:

$$\begin{array}{ll} \min & \frac{1}{2}t + C \bullet X \\ \text{s.t.} & \begin{bmatrix} \mathcal{A} \\ \mathcal{R} \\ 0 \\ 0 \\ 0 \end{bmatrix} X + \begin{bmatrix} 0 & 0 \\ 0 & -I \\ 0 & 0 \\ 0 & 0 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} s \\ z \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -1 \\ -1 \\ 0 \end{bmatrix} t = \begin{bmatrix} b \\ 0 \\ 1 \\ -1 \\ 0 \end{bmatrix} \\ & X \succeq 0,\ \|z\| \le s,\ \|[v;w]\| \le u,\ t \ge 0. \end{array} \tag{11}$$

Appendix: An infeasible primal-dual interior-point algorithm

Here we give a **pseudo-code** for the algorithm we implemented.

Algorithm QSDP-IPC. Suppose we are given an initial iterate (X^0, y^0, Z^0) with $X^0, Z^0 \succ 0$. Set $\gamma^0 = 0.9$.
For $k = 0, 1, \ldots$

(Let the current and the next iterate be (X, y, Z) and (X^+, y^+, Z^+) respectively. Also, let the current and the next step-length parameter be denoted by γ and γ^+ respectively.)

- Compute $\mu(X, Z)$ as in (7).
- (Convergence test)
  Stop the iteration if the accuracy measure ϕ defined in (8) is sufficiently small.
- (Predictor step)
  Compute the predictor direction $(\delta X, \delta y, \delta Z)$ from (4)–(5) by choosing $\sigma = 0$ in R_c.
- (Predictor step-length)
  Compute

$$\alpha_p \ = \ \min\left(1, \tau\,\alpha\right). \tag{12}$$

  Here α is the maximum step length that can be taken so that $X + \alpha\delta X$ and $Z + \alpha\delta Z$ remain positive semidefinite.
- (Centering parameter)
  Set $\sigma = \mu(X + \alpha_p\delta X, Z + \alpha_p\delta Z)/\mu(X, Z)$.
- (Corrector step)
  Compute the correct direction $(\Delta X, \Delta y, \Delta Z)$ from (4)–(5), with R_c in (4) appropriately replaced.

- (Corrector step-length)
  Compute α_c as in (12) but with $(\delta X, \delta Z)$ replaced by $(\Delta X, \Delta Z)$.
- Update (X, y, Z) to (X^+, y^+, Z^+) by

$$X^+ = X + \alpha_c \Delta X, \quad y^+ = y + \alpha_c \Delta y, \quad Z^+ = Z + \alpha_c \Delta Z. \qquad (13)$$

- Update the step-length parameter by $\tau^+ = 0.9 + 0.08\alpha_c$.

References

[1] P. Apkarian, D. Noll, J.-P. Thevenet, and H. D. Tuan, *A spectral quadratic-SDP method with applications to fixed-order H_2 and H_∞ synthesis*, Research Report, Universite Paul Sabaitier, Toulouse, France, 2004.

[2] B. Borchers, *SDPLIB 1.2, a library of semidefinite programming test problems*, Optimization Methods and Software, 11 & 12 (1999), pp. 683–690. Available at `http://www.nmt.edu/~borchers/sdplib.html`.

[3] R.W. Freund and N.M. Nachtigal, *A new Krylov-subspace method for symmetric indefinite linear systems*, Proceedings of the 14th IMACS World Congress on Computational and Applied Mathematics, Atlanta, USA, W.F. Ames ed., July 1994, pp. 1253–1256.

[4] K. Fujisawa, M. Kojima, K. Nakata, and M. Yamashita, *SDPA (SemiDefinite Programming Algorithm) User's manual — version 6.2.0*, Research Report B-308, Department Mathematical and Computing Sciences, Tokyo Institute of Technology, December 1995, revised September 2004.

[5] K. Fujisawa, M. Kojima, and K. Nakata, *Exploiting sparsity in primal-dual interior-point method for semidefinite programming*, Mathematical Programming, 79 (1997), pp. 235–253.

[6] G.H. Golub and C.F. Van Loan, *Matrix Computations*, 2nd ed., Johns Hopkins University Press, Baltimore, MD, 1989.

[7] N.J. Higham, *Computing the nearest correlation matrix — a problem from finance*. IMA J. Numerical Analysis, 22 (2002), pp. 329–343.

[8] M. Kojima, S. Shindoh, and S. Hara, *Interior-point methods for the monotone linear complementarity problem in symmetric matrices*, SIAM J. Optimization, 7 (1997), pp. 86–125.

[9] S. Mehrotra, *On the implementation of a primal-dual interior point method*, SIAM J. Optimization, 2 (1992), pp. 575–601.

[10] M.J. Todd, K.C. Toh, and R.H. Tütüncü, *On the Nesterov-Todd direction in semidefinite programming*, SIAM J. Optimization, 8 (1998), pp. 769–796.

[11] K.C. Toh, M.J. Todd, and R.H. Tütüncü, *SDPT3 — a Matlab software package for semidefinite programming*, Optimization Methods and Software, 11/12 (1999), pp. 545-581.

[12] R.H. Tütüncü, K.C. Toh, and M.J. Todd, *Solving semidefinite-quadratic-linear programs using SDPT3*, Mathematical Programming Ser. B, 95 (2003), pp. 189–217.

[13] K.C. Toh, R.H. Tütüncü, and M.J. Todd, *Inexact primal-dual path-following algorithms for a special class of convex quadratic SDP and related problems.* Pacific J. Optimization, 3 (2007), pp. 135–164

[14] K.C. Toh, *An inexact primal-dual path-following algorithm for convex quadratic SDP*, Mathematical Programming, 112 (2007), pp. 221–254.